(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 109 567**
A2

(12) EUROPEAN PATENT APPLICATION

(21) Application number: 83110498.9

(22) Date of filing: 21.10.83

(51) Int. Cl.³: G 06 F 9/44

(30) Priority: 22.10.82 PC T/US82/01496

(43) Date of publication of application: 30.05.84 Bulletin 84/22

(84) Designated Contracting States: DE FR GB IT

(71) Applicant: International Business Machines Corporation
Old Orchard Road
Armonk, N.Y. 10504(US)

(72) Inventor: Fisk, Dale Edward
1704 Husted Avenue
San Jose California 95124(US)

(72) Inventor: Griffith, Robert Leroy
5995 Alvarado Court
San Jose California 95120(US)

(72) Inventor: Homan, Merle Edward
236 Old Adobe Road
Los Gatos California 95030(US)

(72) Inventor: Radin, George
26 Franklin Street
Piermont New York 10968(US)

(72) Inventor: Richards, Waldo
7167 Via Romero
San Jose California 95139(US)

(74) Representative: Lancaster, James Donald
IBM United Kingdom Patent Operations Hursley Park
Winchester, Hants, S021 2JN(GB)

(54) Accelerated instruction mapping external to source and target instruction streams for near realtime injection into the latter.

(57) If a predetermined field (Figure 3/27) within a source instruction indexes and accesses a body of control information from memory (Figure 2/5), and if control information (Figure 4) designates the field-to-field (register-to-register) mapping (Figure 6), then a skeleton target instruction (Figure 3/29; Figure 4) can be filled in by either selectively copying the fields of the source instruction or otherwise computing same. If the mapping is executed by an interposed independent processor then overlapping of such conversion enhances throughput, the independent processor converting multifield instructions for a CPU of a first kind to multifield instuctions for a CPU of a second kind without disrupting the logical flow or execution of either source or target instruction streams.

EP 0 109 567 A2

ACCELERATED INSTRUCTION MAPPING EXTERNAL TO SOURCE AND TARGET INSTRUCTION STREAMS FOR NEAR REALTIME INJECTION INTO THE LATTER

Technical Field

This invention relates to a new facility realtime format conversion of multifield instructions for a CPU of a first kind to multifield instructions for a CPU of a second kind by facilities external to the CPU of the second kind and without disrupting the logical flow or execution of either source or target instruction streams.

Background Art

Parks et al, U.S. Patent 4,315,321, "Method and Apparatus for Enhancing the Capabilities of a Computing System", issued 9 February 1982, teaches that indicator codes within an activation record can select one of several mutually exclusive microcode sets for interpreting a referenced instruction stream. The activation record is manifest in the form of program status word register bit positions. Parks is concerned with interpreting or construing the intent of each referenced main memory instruction. Further, the activation record of the process external to the instruction provides the microcode set selection index.

Nutter, U.S. Patent 3,543,245 "Computer Systems", issued 24 November 1970, asserts that if control words, indexed by the OP code portion of a CPU multifield instruction, are used to select the instruction fields and their microcode execution order, then varying

instruction widths can be accommodated. Nutter, as does Parks, is concerned with the CPU executing the intent of an external instruction by way of immediate microcode interpretation. Nutter's control words include masks and switching and shifting circuit control bits by which fields in a source instruction would be selected and reordered to form a "queue" for immediate microcode execution. Indeed, the field selection is described from the specification, column 6, line 23, through column 10, line 54, while the mapping of randomly ordered fields in the source word into predetermined positions in a target word is described at column 62, lines 5 through 36. Further, Nutter shows a control word register pair of FIG. 2 as a single register equivalent in FIG. 38. These are discussed at column 24, line 60 through column 31, line 30 in FIGS. 2, 9, and 37.

Other pertinent references include Cassonnet et al, U.S. Patent 3,997,895, "Data Processing System with a Microprogrammed Dispatcher for Working Either in Native or Non-native Mode", issued 14 December 1976, and Malcolm et al, U.S. Patent 3,698,007, "Central Processor Unit Having Simulative Interpretation Capability", issued 10 October 1972. Cassonet depicts a microprogrammable switch (130) responsive to preselected bit position contents in an external instruction for having control stored microcode sequences interpreted respectively by the arithmetic logic unit (ALU 1317) or emulator unit (EMU 1316). Malcolm uses the OP code of the simulated instruction as an index into a set of simulator routines, and provides for storage of a base address to which the OP code index is an offset. Lastly, each instruction references only one operand. This configuration directly executes the intent of the non-native instructions.

A class of VLSI implementable computers with reduced instruction sets being driven by a respective data stream and instruction stream from corresponding caches has been described by Radin in "The 801 Minicomputer", appearing in the ACM Procedings of the Symposium on Architectural Support for Programming Languages and Operating Systems", March 1-3, 1982, in Palo Alto, California, at pages 39-47. A similar CPU architecture was described by Patterson and Sequin in "RISC 1: a Reduced Instruction Set VLSI Computer", in the IEEE 8th Annual Symposium on Architecture Conference Procedings of May 12-14, 1981, at pages 443-449, and in expanded form in IEEE Computer, September 1982 at pages 8-20. In this type of machine, instructions are obtained from an "Instruction Cache", and data is obtained from a separate (data cache), both of which are managed by a LRU information algorithm. Thus, all frequently used functions and data are likely to be found in their respective cache.

Disclosure of the Invention

It is an object of this invention to convert multifield source instructions into multifield target machine instructions and insert them into a target machine instruction stream without otherwise perturbing the normal target machine instruction execution sequence. It is a related object to devise an efficient method of mapping the register space and constants of the source instruction set into that of the target wherein the method does not participate itself in the execution of these instructions. It is still a further object that such a conversion be executed external to the target machine and in near realtime, permitting the target machine to participate in emulations without itself being substantially modified.

The foregoing objects are satisfied by a method for transforming source instructions ordinarily executable by a first CPU-type (source machine) into one or more instructions (code words) to be directly injected into the executable code stream of a second CPU-type (target machine). The method steps comprise (a) fetching a microinstruction comprising a control section and a skeleton target CPU instruction from a memory at a location addressed by a predetermined field of said source instruction; (b) filling in the skeleton according to the control section contents by copying or computing from selected fields of said source instructions; and (c) inserting the filled-in target instructions into the target machine instruction stream.

The apparatus of the invention includes a first and second register; means for loading a source instruction into said first register; means responsive to the OP code contents within said first register for loading a microinstruction control section (control word) into the second register; mapping logic conditioned by the control word in the second register for selectively copying (gating out) or computing from source instruction fields into the skeleton instruction; and means for merging the "fleshed out" target instruction into the counterpart target CPU instruction stream.

The invention is predicated on a number of unexpected observations. These are (1) if a data stream comprising multiple field source machine instructions is mapped into the instruction stream of the target machine by an interposed independent processor, it enhances full realtime utilization due to the independent overlapping of such conversion; (2) if the preponderance of the source instruction fields can be used unchanged in the target machine instructions, then reformatting within an

independent processor can be implemented by register-to-register transfers; (3) if a predetermined field within a source instruction indexes and accesses a word pair from memory, and if one word of the pair is a control section designating the field-to-field (register-to-register) mapping, and if the other word of the pair is a skeleton target instruction, it can be filled out by the fields of the source instruction; and (4) if target machine instructions are constructed external to said target machine, then the target machine is less complex and admits faster instruction execution.

Brief Description of the Drawings

FIG. 1 depicts the fields of an IBM 370 CPU instruction and its general mapping relation to a target machine instruction;

FIG. 2 depicts the emulator-assist processor (EAP) of the invention in communicating relation with the instruction and data caches of the target machine;

FIG. 3 sets out a bare relation of the source multifield IBM 370 CPU instruction and the microinstruction, including the skeleton target instruction to be fleshed out through the mapping logic contained within the EAP;

FIG. 4 shows a definition of a microinstruction control section used by the EAP in fleshing out a skeleton target machine instruction;

FIG. 5 is a timing diagram of the major reformatting operations of the EAP; and

FIG. 6 is a completed field register definition of the EAP set out in FIG. 3.

Best Mode for Carrying out the Invention

While the invention does not reside in the architecture of either the target or source CPU instruction stream generators or receivers, the target CPU does serve as the environment within which the invention is practiced. As the aforementioned Radin and Patterson references exemplify, the new trend in CPU architecture is the use of a reduced instruction set and of independently pipelined instruction and data streams terminating in said CPU. For many years instruction and data reference speeds have been increased by use of least recently used (LRU) managed information caches between the CPU main memory and the target CPU. Thus, the immediately referenced instruction stream is resident in one cache while the immediate reference data stream is referenced in a second. Such a target CPU is shown in FIG. 2.

Typically, the target machine (CPU) 1 is organized to permit independent memory access for the data and instructions. Each access path is served by an independent cache. Thus, instruction cache 5 is accessed by address line 9 with the information therefrom being read over path 11, 13, and 21. Likewise, data cache 7 is accessed over address line 17 and its contents read by target CPU 1 over path 19. However, during realtime instruction translation, data cache 7 writably terminates instruction streams from source CPUs. This means that the data cache is the node from which the source instruction streams are accessed. In this regard, an IBM System 370 CPU is an illustrative

multifield instruction stream source whose instructions can be locally stored in data cache 7. A complete description of IBM 370 host architecture is set out in G.M. Amdahl et al, U.S. Patent 3,400,371, issued 3 September 1968. The 3,400,371 patent is incorporated by reference.

An apparatus embodiment of the invention is in the form of an emulator assist processor (EAP) 3 accessing data cache 7 by way of address path 17a and read path 19a and the instruction cache 5 by way of address path 9 and read path 11a. The conversion output from the EAP is to target CPU machine 1 over path 15, merge 13, and line 21.

With these factors in mind reference should be made to FIG. 1 depicting the fields of an IBM 370 CPU instruction and its general mapping relations to a target machine instruction. Instructions in the IBM 370 System computers consist of 2, 4, or 6 bytes and can contain up to 3 addresses. Five distinctive formats are used depending on the location of various operands required. The formats include:

1. RR (register/register) instructions. The operands R_1 and R_2 are CPU general registers. The result is placed in R_1.

2. RX (register/index) instructions. A first operand is located in R_1 while the other is in main memory. The effective memory address is $X_2 + B_2 + D_2$ where X_2 and B_2 denote the contents of general registers being used as index and base registers respectively, and D_2 is a relative address or "displacement" contained in the instruction. The result is placed in R_1.

3. RS (register/storage) instructions. Two operands are in general registers, a third is in main memory.

4. SI (storage/immediate) instructions. In this case, one operand is in main memory while the other is located within a predetermined range of contiguous bit positions of the instruction itself. This is an immediate operand as opposed to the usual operand address.

5. SS (storage/storage) instructions. Both operands are in main memory. The address as specified by the instructions are typically the initial addresses of two operand fields whose length is L bytes.

With reference to FIG. 1, the 370 instruction depicts an operation code field, typically of one byte followed by a pair of operands L1, L2 and a pair of base-plus-displacement addresses, namely B_1, D_1 and B_2, and D_2. These are to be mapped into a target machine instruction of 32 bits. The target instruction format includes an OP code field occupying bit positions 0-5, an RT field designating the register used to receive the result of an instruction in the positions 6-10, while the RA field in positions 11-15 is the name of the register used for the first operand. Depending on instruction type, the second half of the instruction could include, in positions 16-20, the name of the register used as a second operand, in positions 21-25 the immediate field specifying the operation to be executed by a controller named in an adjacent field of bit positions 26-29. The remaining bit position contents define internal bus operation instructions.

Referring now to FIG. 3, when taken together with FIG. 2, it is apparent that when data cache 7 is addressed over path 17a, the contents consisting of a source instruction, are transmitted over path 19a and loaded into register 25. The OP code of the source instruction, accesses instruction cache 5 by way of address register 27 actuating path 9a. Responsively a microinstruction control section is transmitted to register 23 over path 11a. Each microinstruction may cause a subsequent microinstruction to be accessed so that each source instruction is replaced by an EAP microcode routine. A microcode instruction consists of a control section and a skeleton target instruction. The skeleton target instruction may have zeroed and/or meaningful register and displacement fields. Control information specifies how fields from the source instruction should be merged into the zeroed fields of the skeleton instruction by the EAP. During emulation, the EAP passes these completed target instructions to the target CPU to be executed. The target CPU executes these instructions normally, except that its instruction address register (not shown) remains fixed and the target CPU makes no attempt to fetch instructions. This parenthetically is termed cycle stealing. During emulation, the target CPU waits for the EAP to give it instructions to execute instead of fetching instructions itself. One way of terminating the translation for any specific source instruction can be upon EAP detection of a zeroed instruction field or a stop bit embedded in a predetermined bit position within a microcode sequence.

In executing translation, the target CPU initializes the EAP registers 27. A suitable state change is made in the target CPU. The first source instruction is fetched into the EAP internal register 25. The OP code portion of the source instruction forms

the address to the first microcode instruction for this particular source instruction operation. The microcode instruction is then fetched from the instruction cache. The skeleton target instruction portion of the microinstruction has its zeroed fields filled in from the appropriate fields of 370 instructions. The completed target instruction is then sent to the target machine for execution. Each microinstruction may either link to another microinstruction to be so processed, or it may be the last of a series for the current source instruction. This process is singularly repeated for each 370 or source instruction that is fetched. Significantly, each valid target instruction requires a microcode instruction of two words from the instruction cache. These are the control word and the skeleton target instruction. These are fetched consecutively with the OP code selected control word being first.

Referring now to FIG. 4, there is shown the emulator micro control section format. The format of the 32 bits that make up the control section is allocated as follows: OP is the command to be executed in the EAP, R is the substitution control for the RT and RA target machine register fields, D is the substitution control for the displacement field and the RB target machine register field, C controls the condition codes while NI is the address of the next instruction to be executed by the EAP. If NI is 0, then the EAP will fetch and emulate the next System 370 instruction from the data cache, otherwise it will access the instruction cache again according to the content of the NI. This is aptly drawn in the FIG. 6 enhancement of the EAP 3 shown in FIG. 3. Note in the micro instruction formatted at register 23 in FIG. 6, an alternative to a 0 next instruction address for terminating the EAP fetch from the instruction cache 27 can be by way of a LAST bit position which is set when

the last instruction has been fetched in a sequence from the instruction cache.

Referring now to FIG. 5, there is shown a timing diagram of the major reformatting operations of the EAP in overlap relation (pipelining) to increase throughput. While such pipelining is not the object of this invention, it is evident that significant performance throughput can be obtained.

The register transformation technique between the source and target register spaces provides significant performance gains. For example, because of the pipelining and merging of reformatted instructions from the EAP into the target machine instruction stream, the target machine which might normally execute instructions only every other cycle would permit execution to take place every cycle. This permits the EAP to cause repetitive functions to be executed in the target machine at the full execution rate.

Advantageously, the EAP can be operated in a subroutine mode whenever a sequence of source instructions do not require register space mapping. In this mode, the EAP receives regular target machine instructions, instead of microinstructions from the instruction cache and the target machine again runs at full speed. The subroutine mode is terminable when the target machine is asked to execute an instruction which indicates the resumption of translation mode.

The embodiment heretofore described presupposes formation by the EAP of a complete target machine instruction by merging data extracted from the source instruction with the skeleton instruction from the instruction cache. This is illustrated in FIGS. 2, 3 and

6. One modification involves intercepting the skeleton target instruction and substituting fields in the complete target machine instruction before passing it into the target CPU, rather than merging it on the fly.

While the invention is particularly described with reference to a preferred embodiment it is to be appreciated that the method focuses on dynamic register field substitution on the fly. Source instruction strings generated from CPU's other than the IBM System 370 are certainly contemplated.

In order to avoid EAP bottlenecking, a multiple cache target machine is desirable for performance advantages.

THE CLAIMS

1. A method for converting a source CPU multifield instruction obtained from memory as data into one or more target CPU multifield instructions to be directly injected into the executable code word stream of a target CPU type; comprising the steps of:

fetching a microinstruction and at least one skeleton target instruction from the memory at a location indexed by a predetermined field of said source instruction;

filling in the skeleton target instruction according to the microinstruction contents by copying or computing from selected fields of said source instruction into the skeleton instruction; and

inserting said filled-in target instruction into the CPU instruction stream.

2. A method according to claim 1 wherein each fetched microinstruction includes an address portion which, if non-zero, designates a memory location of a successive microinstruction to be fetched from the memory or, if zero, indicates termination of the microinstruction code word sequence.

3. A method according to claim 1 wherein the steps of fetching, filling-in, and inserting are performed in time overlap relation.

4. A translator for use with a dual cache processing unit for converting instructions stored in the first of two caches as data into counterpart code words executable by the processing unit, the translator comprising:

means for fetching an instruction from the first cache;

means for fetching a microinstruction and a skeleton code word from the second cache at a location determined by the operating code portion of the fetched instruction; and

means for filling in the skeleton code word according to the microinstruction with the fields of the fetched instruction and applying the filled-in code word to the processing unit for execution.

5. A translator according to claim 4, characterized in that the fetching means include means for cycle steal accessing the second cache and cycle steal access to instruction execution cycles of the processing unit.

6. An apparatus for format-converting multi field source instructions stored in a data cache into target instructions and inserting them into an instruction stream obtained from an instruction cache without otherwise perturbing target machine instruction execution, comprising:

a first and a second register;

means for accessing the data cache and loading a source instruction into said first register, said source instruction including an OP field;

means responsive to the OP field contents within said first register for cycle steal accessing the instruction cache and loading a control word into the second register;

mapping logic conditioned by the control word in the second register for selectively copying or gating out source instruction fields from the first register into the skeleton instruction; and

means for cycle stealing the target machine and merging the formatted target instruction into the counterpart instruction stream.

7. In combination with at least one source and one target CPU of dissimilar executable instruction formats, an apparatus for translating multifield instructions from the source into a target CPU format and for injecting the translated instructions into the directly executable target CPU instruction stream, each source instruction having at least one OP field, comprising:

memory means;

means responsive to each multifield source instruction for fetching a body of control information and skeleton target instructions from said memory means at a location indexed by the source instruction OP field;

means for filling in the fields of the fetched skeleton target instructions by either selectively copying source instruction fields or otherwise computing their contents according to the fetched body of control information; and

means for merging the filled in target instructions into the executable instruction stream of the target CPU.

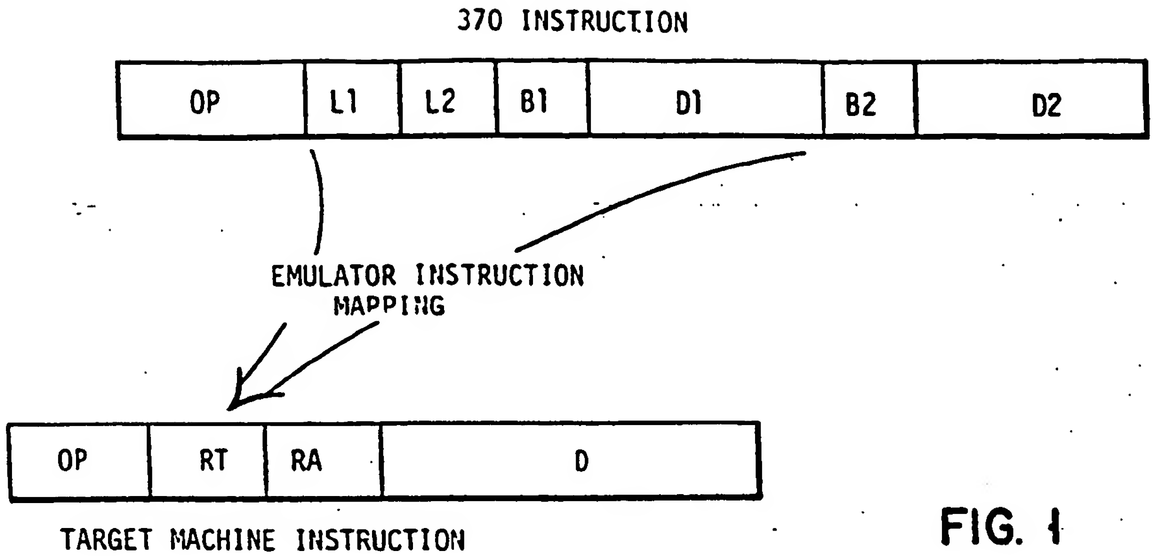
370 INSTRUCTION
OP
L1
L2
B1
D1
B2
D2
EMULATOR INSTRUCTION MAPPING
OP
RT
RA
D
TARGET MACHINE INSTRUCTION

FIG. 1

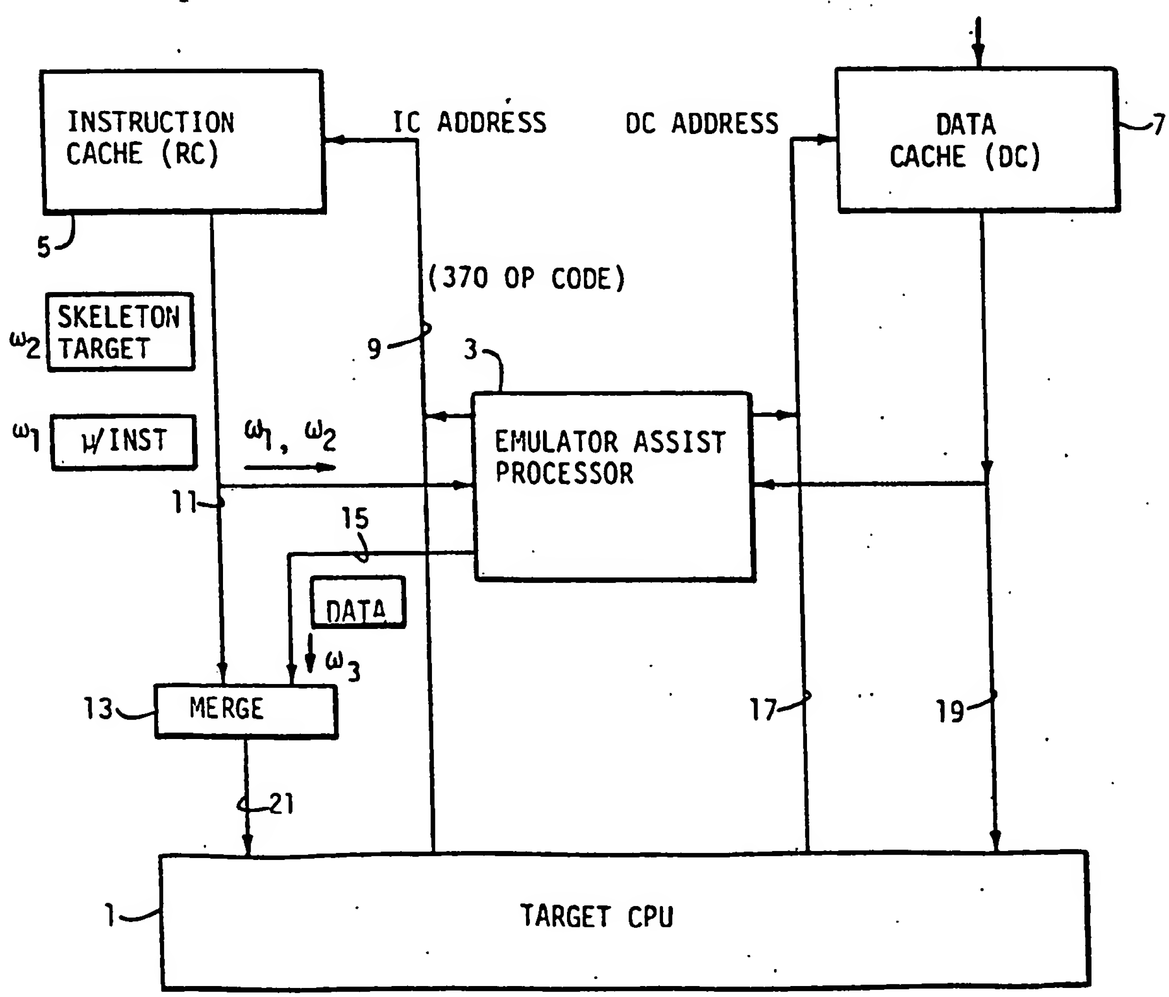
INSTRUCTION CACHE (RC)
5
IC ADDRESS
DC ADDRESS
DATA CACHE (DC)
7
(370 OP CODE)
SKELETON TARGET
ω_2
9
3
ω_1
μ/INST
ω_1, ω_2
EMULATOR ASSIST PROCESSOR
11
15
DATA
ω_3
13
MERGE
17
19
21
1
TARGET CPU

FIG. 2

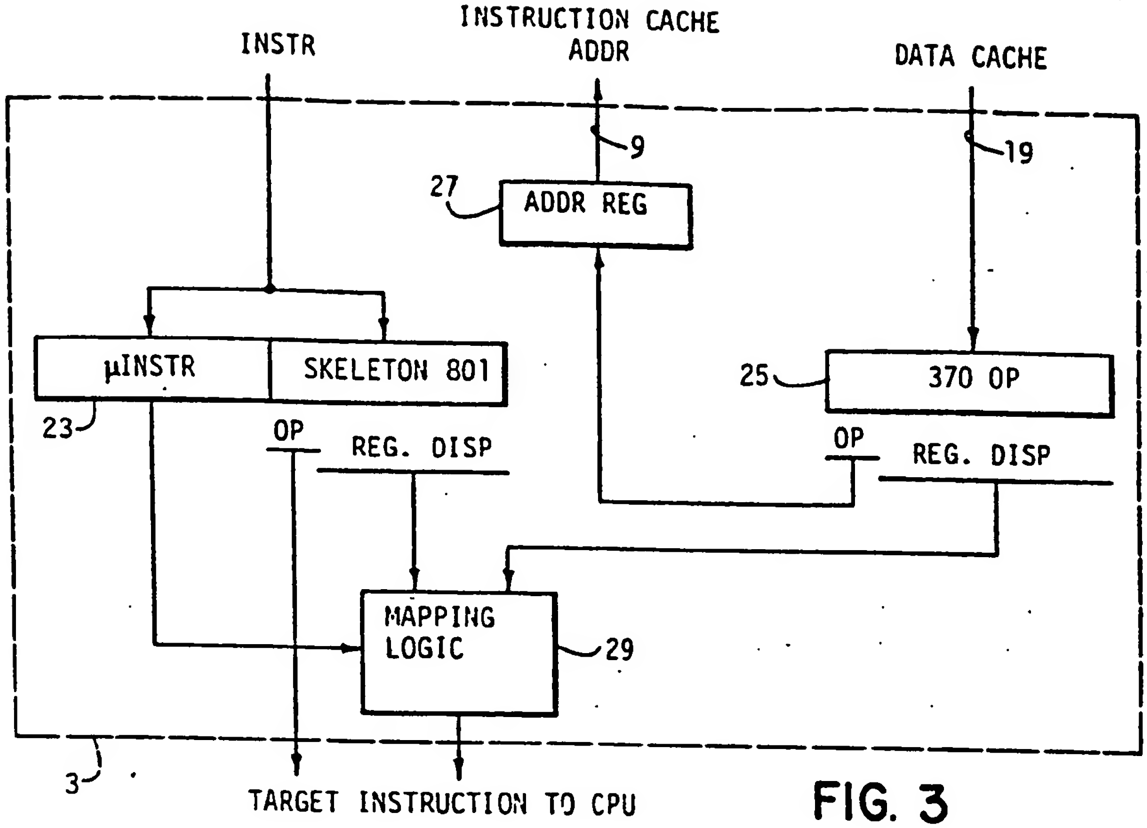

FIG. 3

370 EMULATOR MICROINSTRUCTION

OP	R	D	C		NI
0 – 7	8 – 11	12 – 15	16 – 17	18 – 19	20 – 31

OP	EMULATOR COMMAND
R	SUBSTITUTION CONTROL FOR TARGET MACHINE RT AND RA REGISTER FIELDS
D	SUBSTITUTION CONTROL FOR TARGET MACHINE RB AND DISPLACEMENT FIELDS
C	CONDITION CODE CONTROL
NI	NEXT INSTRUCTION ADDRESS

FIG. 4

OPERATION	MACHINE CYCLE 1	2	3	4	5	6	7	8
FETCH 370 OP	A							
FETCH μ INSTRUCTION			A		B		C	
FETCH 801 SKELETON				A		B		C
MAP 370 TO TARGET MACHINE					A		B	
EXECUTE TARGET MACHINE CODE WORD						A		B

FIG. 5

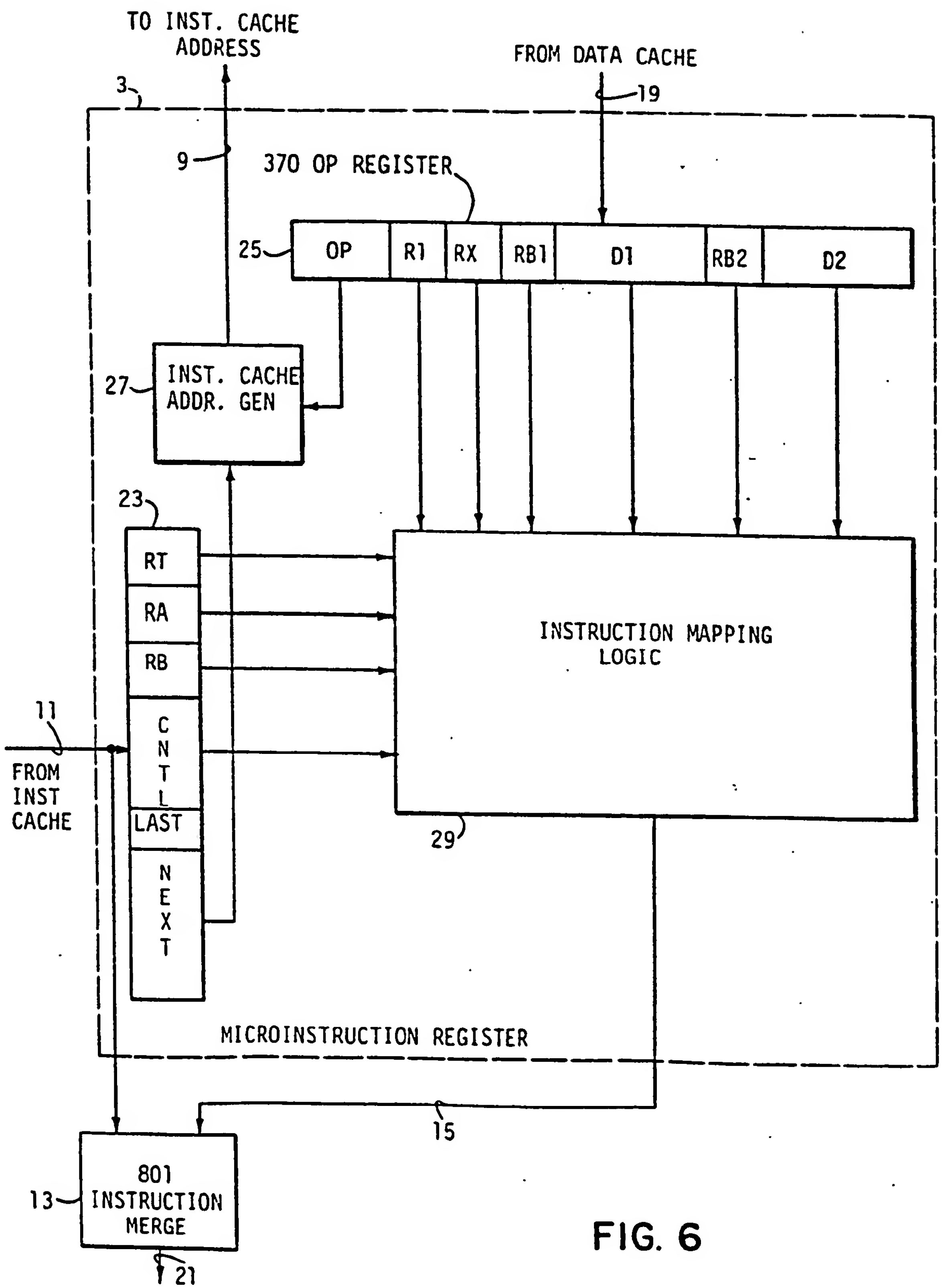
TO INST. CACHE ADDRESS
FROM DATA CACHE
3
19
9
370 OP REGISTER
25
OP
R1
RX
RB1
D1
RB2
D2
27
INST. CACHE ADDR. GEN
23
RT
RA
RB
11
C N T L
LAST
N E X T
FROM INST CACHE
INSTRUCTION MAPPING LOGIC
29
MICROINSTRUCTION REGISTER
15
13
801 INSTRUCTION MERGE
21

FIG. 6